(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 863 645 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
09.09.1998 Bulletin 1998/37

(51) Int Cl.6: H04L 29/00, H04L 12/24

(21) Application number: 98400517.3

(22) Date of filing: 04.03.1998

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 05.03.1997 JP 50074/97

(71) Applicant: KOKUSAI DENSHIN DENWA CO., LTD
Shinjuku-ku Tokyo (JP)

(72) Inventors:
- Yoshihara, Kiyohito
Kokusai Denshin Denwa Co. Ltd.
Shinjuku-ku, Tokyo (JP)
- Horiuchi, Hiroki,
Kokusai Denshin Denwa Co. Ltd.
Shinjuku-ku, Tokyo (JP)
- Sugiyama, Keizo,
Kokusai Denshin Denwa Co. Ltd.
Shinjuku-ku, Tokyo (JP)
- Obana, Sadao, Kokusai Denshin Denwa Co. Ltd.
Shinjuku-ku, Tokyo (JP)

(74) Representative: Lemoine, Robert et al
Cabinet Malémont
42, Avenue du Président Wilson
75116 Paris (FR)

(54) Method for access control of MIB in OSI management

(57) The managed object instance is used as access control unit, then access denial and permission are quickly decided. After a pre-process to the naming tree which is an object of access control, MOI included within every scope which can be designated by the management operation is obtained. At every issue of the management operation, access denial and permission are decided by using the data obtained in the pre-process. When the configuration of the naming tree is changed, the data are easily and rapidly revised, then the access control is adaptive to the dynamic change of the naming tree.

EP 0 863 645 A2

Description

FIELD OF THE INVENTION

This invention relates to a method for access control on MIB (Management Information Base) in OSI (Open System Interconnection) Management. More especially, this invention relates to an access control method in which a management object instance is used as an access unit. Furthermore, this invention relates to a method for converting an identification name of MOI (Management Object Instance), a method for enumeration of some scope of MOI, a method for enumeration of a target of MOI and a method for detecting an intersection of MOI.

BACKGROUND OF THE INVENTION

Words on a tree, a network management based on an OSI management, MOI and a name tree, scope and target, and an abstract of ITU-T recommendation on access control method are described.

(Words on a tree)

Words such as "parent" on a tree are described.

A set having one or more top point is called as a tree if following conditions①and②are satisfied.

①A set T has a specific top point called as a root.
②A set T of tops except for the root is divided to a vacant set or one or more trees T_1 ..., T_m which have no common set each other. These sets are called as a directly partial tree.

A root of a tree having no directly partial tree is called as a leaf. A top point which is not a root and not a leaf is called as a inner point. FIg.19 shows a tree T which has nine top points indicated by circles 0~8.

In Fig.19, the top point 0 is a root. There are two directly partial trees T_1, T_2 in the tree T. One directly partial tree T_1 comprises one top point 1, another directly partial tree T_{21} comprises top points 2,3,4,5,6,7 and 8. Because the tree T_1 has no directly partial tree, the root 1 of the tree T_1 is a leaf of the tree T.

A top point which is included in the directly partial tree of the tree T of which root is a top point v is called as a descendant of the top point v and the root of the directly partial tree is called as a child of the top point v. The point V is a parent of the child. In Fig.19, Descendants of the point 2 are the points 3 ~8, child of the point 2 are the points 3~5. A parent of the points 3~5 is the point 2.

A length a rout from the root to each point is called as a level of the point and a maximum length among these routs is called as a depth of the tree T. In Fig.19, Length of the rout from the root to the leaf 6 or 7 or 8 is maximum, the depth is 3.

In the table 1, a type of each point, parent, child, descendant and level of the tree T are shown in Fig.19.

(table 1)

top point	type	parent	child	descendant	level
0	root	none	{1,2 }	{1~8}	0
1	leaf	0	{}	{}	1
2	inner point	0	{3~5}	{3~8}	1
3	inner point	2	{6~8}	{6~8}	2
4	leaf	0	{}	{}	2
5	leaf	0	{}	{}	2
6	leaf	0	{}	{}	3
7	leaf	0	{}	{}	3
8	leaf	0	{}	{}	3

(Network management based on OSI management)

In a network management system based on the Open System Interconection (OSI), an abstractly described man-

agement object is defined as some MO (Managed Object) and an information of the MO is exchanged between a manager system and an agent system by using CMIS (Common Management Information Service). See [ITU-T. Rec. X.711, Common Management Information Protocol for ITU-T Applications, Mar. 1991] and [Hisao Ohkane, TCP/IP and OSI network managment~SNMP and CMIP ~, Software Research Center, 1993]. Hereinafter, the management system is called as a manager and the agent system is called as an agent.

Fig.1, shows a network management based on OSI management. In Fig.1, The network management is by a network management system and a managed apparatus. The network management system comprises a management console 11 and a manager 12. The managed apparatus comprises an agent 13 and an MIB (Management Information Base) 14. In the MIB, a group of MO such as total number of packets to be transferred, total number of received packets and total number of received packets including error are stored. The network management is achieved by exchanging a management information on MO through a network 15 between the manager 12 and the agent 1 3, on the basis of using the CMIP (Common Management Information Protocol).

For example, when the manager 12 issued a management operation 16 that means "get" of a number of already received packets, the agent 13 sends a response 17 such as "88 packets" from content of the MIB 14.

(Managed object instance and name tree)

Regarding to MO, a kind of Mo having same character is called as MOC (Managed Object Class). Each instance belong to a certain MOC is called as an MOI (Managed Object Instance). For example of MOC, a printer MOC 18 is shown in Fig.2(A) and a printer MOI 19 in the printer MOC 18 is shown in Fig.2(B).

Regarding to a naming tree, in Fig. 3, the logical naming tree comprises a plural number of MOI 20 shown by white circles. A group of MOI is managed by a tree construction and stored in the MIB. For example of the naming tree, the naming tree 22 of a telecommunication carrier 21 indicated by [XXX].

(Scope and Filter)

In CMIS, there are some scope (scope parameter) and some filter (filter parameter) by which one management operation enables to operate a plural number of MOI for reducing a number of telecommunication between the manager and the agent. Generally, scope and filter are set by an operator and an application program.

Scope is a parameter for designating a range of MOI to be managed in the naming tree. When using scope, BOI (Base Object Instance) is designated, wherein BOI is a start point in the designation of the range. Table 2 shows four kinds of scope defined by CMIS, namely BaseObject scope, BaseToNthLevel scope (N is not a minus integer), NthLevelOnly scope (N is not a minus integer) and WholeSubtree scope. Fig.4 shows some scope. In Fig.4, BOI is MOI 23 indicated by a black circle.

(table 2)

scope	definition
BaseObject	A range is only BOI.
BaseToNthLevel	A range is a group of all MOI from BOI to Nth level MOI. BOI itself is included.
NthLevelOnly	A range is a group of MOI just below Nth level from BOI.
WholeSubtree	A range is a group all MOI below BOI. BOI itself is included.

Namely,

1. As shown in Fig.4(A), an object of the management operation of BaseToNthLevel scope is only BOI 23.
2. As shown in Fig.4(B), objects of the management operation of BaseObject scope are BOI 23 and a group of all MOI from BOI 23 to Nth level (in Fig.4(B), N=2) MOI.
3. As shown in Fig.4(C), objects of the management operation of NthLevelOnly scope are only a group of MOI just below Nth level (in Fig.4(C), N=3) MOI from BOI 23.
4. As shown in Fig.4(D), objects of the management operation of WholeSubtree scope are BOI 23 and a group of all MOI below from BOI 23.

Filter is a parameter for designating further an object of a management operation from the MOI group in the range designated by scope. Filter is a logical equation indicating a size of MOI, coincidence of MOI and existence of MOI itself. For example of a filter using an attribute of Printer MOI 19 shown in FIG.2, there is a filter that (connection

interface = RS232C) and (a number of printed sheets before last one hour > 50), wherein "and" is a logical product.

(Abstract of access control based on ITU-T recommendation X.711)

For an interconnection among telecommunication carriers, the network management based on OSI management is opened and security function such as an access control is very important. In ITU-T recommendation X.71 1, "initiators" MOC, "targets" MOC and "rule" MOC are described and a plan for deciding denial and permission of the access. See [ITU-T. Rec. X.711, System Management : Object and attributes for access control, De c. 1995].

Namely,

1. The "initiators" MOC is an MOC which indicates an initiator (an origin of issue of a management operation).
2. The "targets" MOC is an MOC which indicates an MIB to be protected or to be opened against a certain authority. An object to be protected and an object to be opened are called as a target. The target is designated by scope and filter.
3. The "rule" MOC is an MOC which indicates five rules for deciding denial and permission of the access from the "initiators" MOC and " targets" MOC.
4. As shown in Fig.5, as five rules of "rule" MOC, there are a global denial rule which denies an access of the management operation to all object, an item denial rule which denies an access of the management operation to some object, a global permission rule which permits an access of the management operation to all object, an item permission rule which permits an access of the management operation to some object and an default rule which is applied when it is impossible to decide denial and permission by before-mentioned four rules.
5. Decision of denial and permission is done according to a process shown Fig.5. In the step S1, it is judged whether a global denial rule to be applicable exists or not. If the rule exists, all of access are denied. If the rule does not exist, in the next step S2, it is judged whether an item denial rule to be applicable exists or not. If the rule exists, an access according to an access unit is denied. The access unit will be described after. If the rule does not exist, in the next step S3, it is judged whether a global permission rule to be applicable exists or not. If the rule exists, all of access are permitted. If the rule does not exist, in the next step S4, it is judged whether an item permission rule to be applicable exists or not. If the rule exists, an access according to an access unit is permitted. If the rule does not exist, in the next step S5, an access permission or an access denial is decided by the default rule. The default rule, generally, is set so as to deny the access.

As access unit, there are a management operation (a rough access unit), an MOI being an object in a management operation (a moderate access unit) and an attribution of an MOI being an object in a management operation (a fine access unit). In the case of any access unit, an algorism is necessary to decide denial and permission, wherein the algorism decides an intersection between an object of management operation and the protect object, or decides an object of management operation included within the open object.

However, such algorism is not prescribed by an ITU-T recommendation X.711 at all.

Prior art will be described.

(access control by using a management operation as the access unit)

There is known access control by using a management operation as the access unit reported by [Ohno, Yoda, Fujii ; Access Control Method in Telecommunication Network, CS94(39):19-24, Jun. 1994].

This prior art will be described referring to Fig.6 and table 3. The naming tree T shown in Fig.6 is comprises MOI indicated by A~N. Corresponding to the naming tree T, as shown in table 3, "initiators" MOC, "targets" MOC and "rule" MCC are defined. MOI_A, MOI_B, MOI_c · · · MOI_N are used, in the case of designating each MOI.

(table 3)

MOC	MOI
initiators	X
initiators	Y
targets	MOI_C ,MOI_F, MOI_G, MOI_J : targets 1
targets	MOI_D ,MOI_F , MOI_G : targets 2
rule	X can not access to targets 1. (item denial rule : rule 1)

(table 3) (continued)

MOC	MOI
rule	Y can access to targets 2. (item permission rule : rule 2)
rule	All management operation are denied. (default rule : rule 3)

table 3, the initiators X and the initiators Y are defined as MOI belonged to the "initiators" MOC. The initiators X is MOI indicating the origin X of issue of the management operation and the initiators Y is MOI indicating the origin Y of issue of the management operation. Further, the targets 1 and the targets 2 are defined as MOI belonged to "targets" MOC. The targets 1 is MOI of which protect object and open object are $MOI_{C.}$ $MOI_{F.}$ MOI_G and MOI_J . The targets 2 is MOI of which protect object and open object are $MOI_{D.}$ MOI_F and MOI_G. The rule 1, the rule 2 and the rule 3 are defined as MOI belonged to "rule" MOC. The rule 1 is an item denial rule which denies any management operation from the origin X of issue, the rule 2 is an item permission rule which permits all management operations from the origin Y of issue and the rule 3 is a default rule which denies any management operation from all origin of issue.

(Decision of access denial in Fig.6 and table 3 : process of item denial rule)

For example, if a management operation having "WholeSubtree scope" of which BOI is MOI_J from initiator X the item denial rule 1 is applied according table 3. At this time, as shown in Fig.7, because MOI_J in the management operation 24 is included within protect object 25, the management operation is denied.

Therefor, in the case of using the management operation as an access unit, if there is an intersection between a part of the object of the management operation and the protect object, the management operation is denied.

(Decision of access permission in Fig.6 and table 3 : process of item permission rule)

For example, if a management operation having "2ndLevelOnly scope" of which BOI is MOI_A from initiator Y, the item permission rule 2 is applied according table 3. At this time, as shown in Fig.8, because MOI_E in the management operation 26 is not included within open object 27, the management operation is not permitted.

Therefor, in the case of using the management operation as an access unit, if all the object included within open object, the management operation is not allowed.

As mentioned-above, in the prior art access control using the management operation as an access unit, if there is an intersection between object of the management operation and the protect object, MOI to which access is not permitted occurs even if the access should not been denied.

Further, in the prior art access control using the management operation as an access unit, if there is an intersection between object of the management operation and the protect object, MOI to which access is denied occurrs even if the access is permitted.

These problems do not occur in an access control using the MOI as an access unit.

Then, an object of the present invention is to provide a new access control using the MOI as an access unit.

Another object of the present invention is to provide a method for exchanging the identification name, a method for scope enumeration, a method for target enumeration and a method for detecting an intersection.

SUMMARY OF THE INVENTION

In the present invention, a pre-process is provided for reducing a time require to the denial and permission of access rather than the prior art. In this pre-process, a corresponding table, which indicates a relation between scope and a set of MOI included to the scope. Then, at every issue of the management operation, an intersection between the management operation and the protect object is decided by referring to the table. Further, at every issue of the management operation, a management operation included to the open object is obtained by referring to the table, then access denial and access permission are rapidly decided.

Namely, in the present invention, an identification name of MOI on the naming tree is exchanged to an index. The present invention is a method for converting a name of MOI (Managed Object Instance) in a name tree to an index, wherein "n" denotes a number of MOI in the name tree, "[x]" denotes an integer rounded up from a value x and "XOR" denotes an exclusive OR, said method comprising:

a step for dividing a bit sequence to m blocks Bi ($1 \leqq i \leqq m$), wherein a number of each block is N which is given as a [log2n],

a step for calculating an exclusive OR of a j-th bit b_{ij} ($1 \leqq j \leqq N$) of each block B_j as $C_j = b_{1j}$ XOR b_2, XOR $b_{j3} \cdots$

XOR b_{mj},
a step for making an N bits sequence $C_1C_2C_3 \ldots C_N$ as an index of an identification name by putting said calculated value C_j from C_1 to C_N,

wherein a value "0" is applied to an m-th block B_m if an insufficiency of a bit occurs in the m-th block B_m .

The present invention is a method for enumeration of some scope wherein, regarding to each scope all of which can be designated in a management operation, an arrangement "scope[]" of which size is n and which represents an MOI included in the scope is obtained as,

scope[i]=1 if an MOI_i is included in the scope,
scope[i]=0 if an MOI_i is not included in the scope.

The present invention is a method for enumeration of some scope wherein, when new $MOIm_a$ is added to a name tree, the $MOIm_a$ is added to a BaseTojthLevel scope ($i \leqq j$) of which BOI (Base Object Instance) is an MOI_i ($1 \leqq i \leqq p$), a WholeSubtree scope and an ithLevelOnly scope, wherein MOI on a route from a $MOIm_1$ upper than $MOIm_a$ to a root $MOIm_p$ in the $MOIm_a$ to be added is put in order from $MOIm_1$ as $MOIm_1$, $MCIm_2$, $\cdots$ $MOIm_p$.

The present invention is a method for enumeration of some scope wherein, when an $MOIm_d$ is deleted from a name tree, the $MOIm_d$ is deleted from a BaseTojthLevel scope ($i \leqq j$) of which BOI (Base Object Instance) is an MOI_i ($1 \leq i \leq p$), a WholeSubtree scope and an ithLevelOnly scope, wherein MOI on a route from a $MOIm_1$ upper than $MOIm_d$ to a root $MOIm_p$ in the $MOIm_d$ to be deleted is put in order from $MOIm_1$ as $MOIm_1$, $MOIm_2$, $\cdots$ $MOIm_p$.

The present invention is a method for enumeration of a target wherein, regarding to each targets MOI which is a protect target to be protected from an authority or an open target to be opened to an authority, an arrangement "targets []" of which size is n and which represents the target MOI is obtained as,

targets[i]=1 if an MOI_i is protected or opened,
targets[i]=0 if an MOI_i is not protected and not opened.

The present invention is a method for making a table wherein, regarding to each scope all of which can be designated in a management operation, a table corresponding to an MOI included in the scope is made.

The present invention is a method for detecting an intersection wherein, an intersection between a management object and a protect object is obtained by calculating in each bit a logical product (logical and) between the "scope[]" obtained by any of above-mentioned methods and a denial of the "targets[]" obtained by above-mentioned method.

The present invention is a method for detecting an intersection wherein, an intersection between a management object and a protect object is obtained by calculating in each bit a logical product (and) between the "scope[]" obtained by any of above-mentioned methods and the "targets[]" obtained by above-mentioned method.

The present invention is an access control method by using MOI as an access unit comprises;a step for calculating in each bit a logical product (logical and) between a denial of each "targets[]" in an item denial rule obtained by above-mentioned method and the "scope[]" obtained by any of above-mentioned methods, and a step for allowing only an MOI of which scope[i]=1 based on said calculation.

The present invention is an access control method by using MOI as an access unit comprises;a step for calculating in each bit a logical product (logical and) between each"targets[]" in an item permission rule obtained by above-mentioned method and the "scope[]" obtained by any of above-mentioned methods, and a step for allowing only an MOI of which scope[i]=1 based on said calculation.

These access control are adaptive to a dynamic change of the naming tree based on the management operation such as M-CREATE and M-GET. Namely, based on the management operation such as M-CREATE and M-GET, a new MOI is generated or added to the naming tree, or, an old MOI is deleted from the naming tree. Therefor, it is necessary to renewal the corresponding table. In the present invention, it is possible to easily revise a part to be changed, then it is not necessary to change all of the table.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows an abstract of a network management based on OSI management.
Figs. 2(A) and 2(B) show examples of MOC and MOI.
Fig. 3 shows a naming tree.
Fig. 4(A) shows scope.
Fig. 4(B) shows scope.
Fig. 4(C) shows scope.
Fig. 4(D) shows scope.
Fig. 5 shows a process of decision of denial and permission based on ITU-T recommendation X.711.
Fig. 6 shows a naming tree.
Fig. 7 shows a prior art.
Fig. 8 shows a prior art.

Fig. 9 is a flow chart showing a whole of access control based on the present invention.

Fig. 10 is a flow chart showing the pre-process.

Fig. 11 shows a step for conversion of the identification name.

Fig. 12 shows a step of enumeration.

Fig. 13 shows a naming tree for enumeration.

Fig. 14 is a flow chart showing access denial and access permission of access control based on the present invention.

Fig. 15 is a flow chart for renewal of the corresponding table in the case of adding MOI.

Fig. 16 is an example for renewal of the corresponding table in the case of deleting MOI.

Fig. 17 is a flow chart for renewal of the corresponding table in the case of deleting MOI.

Fig. 18 is an example for renewal of the corresponding table in the case of deleting MOI.

Fig. 19 shows a naming tree.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

An embodiment of the present invention will be explained referring to the drawings.

As shown in Fig.9, in a step S0, a pre-process is carried out to the naming tree which is an object of access control for achieving a rapid access denial and permission. In the pre-processing, at every scope all of which is designated by the management operation, all MOI included within the scope. One time of the pre-processing is sufficient.

After pre-processing, in a step S1, a management operation is issued. Then, in a step S2, access denial and access permission are decided by using a corresponding table which was made in a step for table making in the pre-processing.

In a step S3, it is judged whether a generation or deletion of MOI. If MOI is changed, a renewal of the corresponding table is carried out before next decision of access denial or access permission of the management operation.

(Abstract of pre-processing)

As shown in Fig.10, in the pre-processing, an identification name of MOI is converted in a step S1. After the step S1, at every scope all of which is designated by the management operation, all MOI included within the scope in a step S2. Then, in a step S3, the corresponding table which corresponds to a relation between scope and a set of MOI included within the scope.

(Detailed description 1 of pre-processing : conversion step of identification name)

The identification name of MOI is encoded according to BER (Basic Encoding Rules) of ASN.1 (Abstract Syntax Notation.1) etc.. See [ITU-T. Rec. X.690, ASN.1 encoding rules: Specification of BER, Canonical ENcoding rules (CER), and Distinguished encoding rules (DER), 1994].

The identification name is converted, as shown Fig.11, by allotting an index to the encoded identification name of MOI.

In Fig.11, "r" denotes the identification name comprising "1" and "0" of input MOI, " I r I " denotes a bit length of the identification name "r". "n" denotes a number of MOI in the naming tree, "N" denotes a number of bits of the index allotted to the identification name "r". "[x]" denotes an integer rounded up from a value x. "XOR" denotes some exclusive OR as shown in an equation 3.

(equation 3)

$$X,y \in \{0,1\}$$

$$x \text{ XOR } y = 0 \text{ (when } x=y)$$

$$x \text{ XOR } y = 1 \text{ (when } x \neq y)$$

In the steps shown in Fig.11,

(1) A conversion is started by an input of MOI having the identification name r.

(2) In a step S1, a value N is calculated by $N=[\log_2 n]$ from the number n of MOI in the naming tree. Namely, it is possible to indicate all of MOI by an index which is N bits sequence.

(3) In a step S2, the identification name r is divided to m blocks Bi ($1 \leqq i \leqq m$), wherein a number of each block is N bits. A value "0" is applied to an m-th block B_m if an insufficiency of a bit occurs in the m-th block B_m.
(4) In a step S3, an exclusive OR of a j-th bit b_{ij} ($1 \leqq j \leqq N$) of each block B_j is calculated as $C_j = b_{1j}$ XOR b_{2j} XOR $b_{3j} \cdots$ XOR b_{mj},
(5) The identification name r is converted to an N bits sequence $C_1C_2C_3 \cdots C_N$ by using C_j and the N bits sequence $C_1C_2C_3 \cdots C_N$ is outputted as an index. Namely, the index is made by putting said calculated value C_j from C_1 to C_N in order.
(6) The index allots 0~n-1 in decimal to n MOI on the naming tree. There is not same index among MOI because of using XOR. While the length of the identification name r is not constant among MOI on the naming tree, it is possible to use an index having a constant length because of the conversion of $C_1C_2C_3 \cdots C_N$. Further, it is possible to achieve a high speed access the length of the index $C_1C_2C_3 \cdots C_N$ shorter than the length of the identification name r.

An example of the conversion of the identification name, wherein input identification name r is 10001000 00001110 10110001 00010000 11000100 00011000 and a number n is 100.

(1) In the step S1, $N=[\log_2 100]=[6.6438 \cdots]=7$.
(2) In the step S2, $|r|=48$, m=7 from (48/7)+1, then the identification name r is divided to 7 blocks Bi ($1 \leqq i \leqq 7$), wherein a number of each block is 7 bits. A value "0" is applied to the 7-th bit of final block B_7 because $N \times n-|r|=49-48=1$. Namely,
B_1=1000100, B_2=0000011, B_3=1010110, B_4=0010001, B_5=0000110,
B_6=0010000, B_7=0110000
(3) For example, in the 1st bit b_{i1} of each block B_i ($1 \leqq i \leqq N$), the 1st bit C_1=1 XOR 0 XOR 1 XOR 0 XOR 0 XOR 0 XOR = 0 because of $b_{11}=1$, $b_{21}=0$, $b_{31}=1$, $b_{41}=0$, $b_{51}=0$, $b_{61}=0$, $b_{71}=0$.
(4) In the same way, $C_2=1$, $C_3=0$, $C_4=0$, $C_5=1$, $C_6=1$, $C_7=0$.
(5) Then the identification name r is converted to "0100110" based of the bit sequence $C_1C_2C_3C_4C_5C_6C_7$=0100110.

(Detailed description 2 of pre-processing : enumeration step)

In the enumeration step, as shown in Fig.12, MOI included within scope is obtained at every scope of all scope designated by the management operation. Therefor, as defined by equations 4~7, a matrix A having a size of n x n and a matrix C having a size of n x n. The matrix A is a connection matrix which denotes the naming tree. In a step12 shown in Fig.12, while the BaseTo(i-1)thLevel scope becomes to a BaseTo0thLevel scope, the BaseTo0thLevel scope is treated as a BaseObject scope.

(equation 4) $(a_{ij})=1$, when MOI_i with index i is a parent of MOI_j with index j on the naming tree T,

$(a_{ij})=0$, when MOI_i with index i is not a parent of MOI_j with index j on the naming tree T,

wherein (a_{ij}) is an element on i-th line and j-th row of the matrix A.

(equation 5) $$A^0=E \text{ (unit matrix)}$$

(equation 6) $$A^i = A^* A^{(i-1)} \quad (i \geq 1)$$

(equation 7) $$C^i = A^0 + A^1 + A^2 + \cdots + A^i$$

In fig.12,

(1) In a step S1, A^X and C^X ($1 \leq x \leq D$) are calculated until $A^{n+1}=0$, wherein D is depth of the naming tree T.
(2) In step S2, i and A^i are initialized as i=1 and $A^i = A^* A^{(i-1)} = A$.
(3) In a step S3, when MOI having index j is indicated as MOI_j, j is initialized as j=0.
(4) In a step S4, it is judged whether MOI_j satisfies a condition indicated in next step S5. If not satisfied, the step

S5 is done. If satisfied, a step S8 is done.

(5) In the step S5, when ($a^i{}_{jk}$) indicates an element in j-th line of A^i or an element of which line is MOI_j of a certain matrix A^i, it is judged whether ($a^i{}_{jk}$) is 0 about all k. If not satisfied, the step S6 is done. If satisfied, a step S12 is done.

(6) In the step S6, it is judged that ithLevelOnly scope of which BOI is MOI_j includes MOI_k, wherein ($a^i{}_{jk}$)=1. Then a step S7 is done.

(7) In the step S7, when ($C^i{}_{jk}$) indicates an element in j-th line of C' or an element of which line is MOI_j of a certain matrix C^i, it is judged that BaseToithLevel scope of which BOI is MOI_j includes MOI_k, wherein ($C^i{}_{jk}$)=1. Then a step S8 is done.

(8) In the step S12, WholeSubtree scope of which BOI is MOI_j is treated as BaseTo(i-1)thLevel scope. Then a step S8 is done.

(9) In the step S8, j is increased by 1. Then a step S9 is done.

(10) In the step S9, it is judged whether j is smaller than n. If true, the step S4 is done. If false, a step S10 is done.

(11) In the step S10, i is increased by 1, namely the matrix A^i is changed into a matrix A^{i+1} and the matrix C^i is changed into a matrix C^{i+1}.

(12) In an step S11, it is judged whether i is smaller than D+1. If true, the step S3 is done. If false, the enumeration is finished.

(example of enumeration)

An example of the enumeration step is explained on the naming tree T shown in Fig.13. The connection matrix A of the tree T is shown in the equation 8. The matrix is started from 0-th line and 0-th row.

(equation 8)

$$A=\begin{pmatrix} 01100000 \\ 00000000 \\ 00011100 \\ 00000011 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \end{pmatrix}$$

(1) In the step S1 shown in Fig.12, A^2,A^3,A^4,C^2 and C^3 are obtained as shown in equations 9~13 based on the equations 6 and 7.

(equation 9)

$$A^2=\begin{pmatrix} 00011100 \\ 00000000 \\ 00000011 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \end{pmatrix}$$

(equation 10)

$$A^3=\begin{pmatrix} 00000011 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \end{pmatrix}$$

(equation 11)

$$A^4=\begin{pmatrix} 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \end{pmatrix}$$

(equation 11)

$$C^2=\begin{pmatrix} 01111100 \\ 00000000 \\ 00011111 \\ 00000011 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \end{pmatrix}$$

(equation 12)

$$C^3=\begin{pmatrix} 01111111 \\ 00000000 \\ 00011111 \\ 00000011 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \end{pmatrix}$$

(2) i is set as i=1 in the step S2 and j is set as j=0 in the step S3.
(3) In the step S4, MOI_j does not satisfy the condition indicated in the step S5. Then the step S5 is done.
(4) In the step S5, $(a^i{}_{jk})$ is not 0 about all k. Then, the step S6 is done.
(5) In the step S6, ith(1st)LevelOnly scope of which BOI is MOI_j (=MOI_0) includes MOI_1 and MOI_2, because of $(a^1{}_{01})=1$ and $(a^1{}_{02})=1$.
(6) In the step S7, BaseToithLevel (=BaseToIstLevel) scope of which BOI is MOI_j (=MOI_0) includes MOI_0, MOI_1 and MOI_2, because of $(C^1{}_{00})=1$, $(C^1{}_{01})=1$ and $(C^1{}_{02})=1$.
(7) In the step S8, j=j+1=0+1-1 and j=1<n=8. The step S4 is done because the judgement in the step S9 is false.
(8) In the step S4, MOI, does not satisfy the condition indicated in the step S5. Then the step S5 is done.
(9) In the step S5, $(a^1{}_{1k})$ is 0 about all k. Then the step S12 is done because the judgement in the step S9 is false.
(10) In the step S12, WholeSubtree scope of which BOI is MOI_j (=MOI_0) is treated as BaseTo(i-1)thLevel (= BaseToOthLevel= BaseObject) scope.
(11) By repeating same process, at every scope, a set of MOI included to scope.

(step for making a corresponding table)

A table indicating a correspondence between scope and MOI included within scope is made by using the above-mentioned results. The table shows a part of the table corresponding to the naming tree T shown in Fig.13. Namely,

all of scope is indicated as a combination of a type of scope and MOI. Regarding to each scope all of which can be designated in a management operation, an arrangement "scope[]" of which size is n and which represents an MOI included in the scope is obtained as,

scope[i]=1 if an MOI_i is included in the scope,

scope[i]=0 if an MOI_i is not included in the scope. Then, the corresponding table is made by obtaining "scope[]" .

(table 4)

BOI	type of scope	0 1 2 3 4 5 6 7
0	BaseObject	1 0 0 0 0 0 0 0
0	BaseTo1stLevel	1 1 1 0 0 0 0 0
0	BaseTo2ndLevel	1 1 1 1 1 1 0 0
0	WholeSubtree	1 1 1 1 1 1 1 1
0	1stLevelOnly	0 1 1 0 0 0 0 0
0	2stLevelOnly	0 0 0 1 1 1 0 0
0	3rdLevelOnly	0 0 0 0 0 0 1 1
1	BaseObject	0 1 0 0 0 0 0 0
2	BaseObject	0 0 1 0 0 0 0 0
2	BaseTo1stLevel	0 0 1 1 1 1 0 0
2	WholeSubtree	0 0 1 1 1 1 0 0
2	1stLevelOnly	0 0 0 1 1 1 0 0
2	2stLevelOnly	0 0 0 0 0 0 1 1

Referring to an example shown in the table 4 and Fig.7 and Fig.8, a process to decide an access denial and permission will be described.

(example of decision for an access denial)

In Fig.7, objects of management operation are J, L,M and N out of all MOI indicated by A ~N on the naming tree T. Therefor, scope[i]=1 denotes that MOI_i is included within scope and scope[i]=0 denotes that MOI_i is not included within scope. Size of scope[] is n (A~N). Then scope[] becomes as below. Further, in the table 3, access to the targets 1 (MOI_C, MOI_F, MOI_G, MOI_J) by the initiator X is denied because of the item denial rule 1. Therefor, targets[i]=1 denotes that MOI_i is protected and targets[i]=0 denotes that MOI_i is not protected. Size of targets[] is n. Then targets[] becomes as below.

```
              ABCDEFGHIJKLMN
scope[]    =  00000000010111
targets[]  =  00100110010000
```

Wherein, scope[]=00000000010111(n=14) is quickly and easily obtained by previously making the corresponding table of the tree shown in Fig.7 in the above-mentioned step, because the object of the management operation shown in Fig.7 can be designated by WholeSubtree scope of which BOI is MOI_J.

By the step S3 shown in Fig.14, a denial of each bit element of " targets []" is a bit sequence of 11011001101111. In each bit, a logical product (logical and) between the "scope[]" (= 00000000010111) and a denial (=11011001101111) of the "targets[]" (=00100110010000) is calculated.

Then scope[] becomes as follows.

```
           ABCDEFGHIJKLMN
scope[] =  00000000000111
```

Namely, access to only MOI_J is denied and access to MOI_L, MOI_M and MOI_N are not denied.

(example of decision for an access permission)

In Fig.8, objects of management operation are D, E,F and G out of all MOI indicated by A~N on the naming tree T. Therefor, scope[i]=1 denotes that MOI_i is included within scope and scope[i]=0 denotes that MOI_i is not included within scope. Size of scope[] is n (A~N). Then scope[] becomes as below. Further, in the table 3, access to the targets 2 (MOI_D, MOI_F, MOI_G, MOI_J) by the initiator Y is permitted because of the item permission rule 2. Therefor, targets[i] =1 denotes that MOI_i is opened and targets[i]=0 denotes that MOI_i is not opened. Size of targets[] is n. Then targets[] becomes as below.

```
               ABCDEFGHIJKLMN
scope[]    =   00011110000000
targets[]=     00010110000000
```

Wherein, scope[]=0001111 0000000(n=14) is quickly and easily obtained by previously making the corresponding table of the tree shown in Fig.8 in the above-mentioned step, because the object of the management operation shown in Fig.8 can be designated by 2ndLevelOnly scope of which BOI is MOI_A.

By the step S4 shown in Fig.14, in each bit, a logical product (logical and) between "scope[]" (= 00011110000000) and "targets[]" (=00010110000000) is calculated.

Then scope[] becomes as follows.

```
           ABCDEFGHIJKLMN
scope[]=   00010110000000
```

Namely, access to MOI_D, MOI_F and MOI_G are permitted.

(renewal of corresponding table)

A renewal of the corresponding table will be described.

As mentioned-above, the configuration of the naming tree may be changed when a new MOI is generated or added and when an old MOI is deleted. Therefor it is necessary to renew the table.

(generation of MOI)

Fig.15 shows a step for renewing the table when a new $MOIm_a$ is added to the naming tree.

(1) In a step S1 in Fig.15, all MOI on a route from a $MOIm_1$ upper than $MOIm_a$ to a root $MOIm_p$ are named as $MOIm_1$, MOIm2, · · · $MOIm_p$ in order from $MOIm_1$.
(2) By a step S2 in Fig.15, in the corresponding table, $MOIm_a$ is added to a BaseTojthLevel scope ($i \leqq j$) , a WholeSubtree scope and an ithLevelOnly scope each of which BOI (Base Object Instance) is an $MOIm_i$ ($1 \leqq i \leqq p$).

(example of generation of MOI)

An example of renewal, when $MOI_8(=m_a)$ is added as a child of $MOI_5(=m_1)$ to the naming tree T shown Fig.16, will be described.

(1)) step S1 : A parent of MOI_5 is a 2nd ($=m_2$) line in a 5th ($=m_1$) row of which value is "1" in the matrix A. In the same way, a parent of MOI_2 is a 0th ($=m_3$) line in a 2nd ($=m_2$) row of which value is "1" in the matrix A. Because MOI_0 is root, $MOIm_1 = MOI_5$, $MOIm_2= MOI_2$ and $MOIm_3=MOI_0$ are root.
(2) Step S2 : $MOI_8(=m_a)$ is added to a BaseTojthLevel scope ($1 \leqq j$) , a WholeSubtree scope and a 1stLevelOnly

scope each of which BOI (Base Object Instance) is $MOIm_1$. MOI_8 (=m $_a$) is added to a BaseTojthLevel scope (2 ≦ j) , a WholeSubtree scope and a 2ndLevelOnly scope each of which BOI (Base Object Instance) is $MOIm_2$. MOI_8(=m $_a$) is added to a BaseTojthLevel scope (3 ≦ j) , a WholeSubtree scope and a 3rdLevelOnly scope each of which BOI (Base Object Instance) is $MOIm_3$. Namely, scope[8]=1 is added to above-mentioned scope, scope [8]=0 is added to other scope.

(deletion of MOI)

Fig.17 shows a step for deleting the table when an old $MOIm_d$ is deleted from the naming tree.

(1) In a step S1 in Fig.17, all MOI on a route from a $MOIm_1$ upper than $MOIm_d$ to a root $MOIm_p$ are named as $MOIm_1$, MOIm2, · · · $MOIm_p$ in order from $MOIm_1$.
(2) By a step S2 in Fig.17, in the corresponding table, $MOIm_d$ is deleted from a BaseTojthLevel scope (i ≦ j), a WholeSubtree scope and an ithLevelOnly scope each of which BOI (Base Object Instance) is an $MOIm_i$ (1 ≦ i ≦ p).

(example of deletion of MOI)

An example of deletion, when MOI_7 which is a child of MOI3 is deleted from the naming tree T shown Fig.18, will be described.

(1) step S1 : A parent of MOI_7 (= m_d) is a 3rd (=m_1) line in a 7th (=m_d) row of which value is "1" in the matrix A. In the same way, a parent of MOI_3 (=m_1) is a 2nd (=m_2) line in a 3rd (=m_1) row of which value is "1" in the matrix A. Because MOI_0 is root, $MOIm_1$ = MOI_5, $MOIm_2$ = MOI_2 and $MOIm_3$=MOI_0 are root.
(2) Step S2 : MOI_7 (=m_d) is deleted from BaseTojthLevel scope (1 ≦ j), WholeSubtree scope and 1stLevelOnly scope each of which BOI (Base Object Instance) is $MOIm_1$. MOI_7 (=m $_d$) is deleted from BaseTojthLevel scope (2 ≦ j), WholeSubtree scope and 2ndLevelOnly scope each of which BOI (Base Object Instance) is $MOIm_2$. MOI_7 (=m $_d$) is deleted from BaseTojthLevel scope (3≦j) , WholeSubtree scope and 3rdLevelOnly scope each of which BOI (Base Object Instance) is $MOIm_3$. Namely, scope[7]=1 is deleted from above-mentioned scope, scope[7]=0 is deleted from other scope.

Above-mentioned process is generally carried out by a computer. Namely, the computer carries out the process by reading data which were programed data of the process and readable data to the computer and stored in a recording medium. Then the computer is an apparatus having a function which carries out above-mentioned process.

(effect of the invention)

The present invention has following effect (1) and (2) than the prior art.

(1) Calculation amount is smaller than the prior art of access control using the management operation as access unit.
(2) Fine access control using the managed instance object as access unit is possible.

Further, by converting to the index from the identification name of MOI, it is possible to unify the bit length of the identification name and quick access to MOI is possible. By the enumeration process, it is possible to simply and clearly indicate MOI which is included within scope which can be designated by the management operation and it is possible to simply and clearly indicate the object to be protected or opened. By renewing the corresponding table, it is possible to simply and easily adapt to a change of MOI which occurs based on the addition or deletion of MOI on the naming tree. Further, by calculating a logical product in each bit of "scope[]" and "targets[]", it is possible to simply and easily obtain the intersection between scope and the protected object or the opened object.

Claims

1. A method for converting a name of MOI (Managed Object instance) in a name tree to an index, wherein "n" denotes a number of MOI in the name tree, "[x]" denotes an integer rounded up from a value x and "XOR" denotes an exclusive OR, said method comprising:

a step for dividing a bit sequence to m blocks Bi (1 ≦ i ≦ m), wherein a number of each block is N which is

given as a $[\log_2 n]$,
a step for calculating an exclusive OR of a j-th bit b_{ij} ($1 \leqq j \leqq N$) of each block B_j as $C_j = b_{1j}$ XOR b_{2j} XOR b_{3j} $\cdots$ XOR b_{mj},
a step for making an N bits sequence $C_1C_2C_3 \cdots C_N$ as an index of an identification name by putting said calculated value C_j from C_1 to C_N,

wherein a value "0" is applied to an m-th block B_m if an insufficiency of a bit occurs in the m-th block B_m.

2. A method for enumeration of some scope wherein, regarding to each scope all of which can be designated in a management operation, an arrangement "scope[]" of which size is n and which represents an MOI included in the scope is obtained as,
scope[i]=1 if an MOI_i is included in the scope,
scope[i]=0 if an MOI_i is not included in the scope.

3. A method for enumeration of some scope wherein, when new $MOIm_a$ is added to a name tree, the $MOIm_a$ is added to a BaseTojthLevel scope ($i \leqq j$) of which BOI (Base Object Instance) is an MOI_i ($1 \leqq i \leqq p$), a WholeSubtree scope and an ithLevelOnly scope, wherein MOI on a route from a $MOIm_1$ upper than $MOIm_a$ to a root $MOIm_p$ in the $MOIm_a$ to be added is put in order from $MOIm_1$ as $MOIm_1$, $MOIm_2$, $\cdots$ $MOIm_p$.

4. A method for enumeration of some scope wherein, when an $MOIm_d$ is deleted from a name tree, the $MOIm_d$ is deleted from a BaseTojthLevel scope ($i \leqq j$) of which BOI (Base Object Instance) is an MOI_i ($1 \leqq i \leqq p$), a WholeSubtree scope and an ithLevelOnly scope, wherein MOI on a route from a $MOIm_1$ upper than $MOIm_d$ to a root $MOIm_p$ in the $MOIm_d$ to be deleted is put in order from $MOIm_1$ as $MOIm_1$, $MOIm_2$, $\cdots$ $MOIm_p$.

5. A method for enumeration of a target wherein, regarding to each targets MOI which is a protect target to be protected from an authority or an open target to be opened to an authority, an arrangement " targets[]" of which size is n and which represents the target MOI is obtained as,
targets[i]=1 if an MOI_i is protected or opened,
targets[i]=0 if an MOI_i is not protected and not opened.

6. A method for making a table wherein, regarding to each scope all of which can be designated in a management operation, a table corresponding to an MOI included in the scope is made.

7. A method for detecting an intersection wherein, an intersection between an management object and an protect object is obtained by calculating in each bit a logical product (logical and) between the " scope[]" obtained in claim 2 or 3 or 4 or 6 and a denial of the " targets[]" obtained in claim 5 which is shown in below equation 1,

$$\text{(equation 1)} \qquad \text{denial of "targets[]"="}\overline{\text{targets[]}}\text{"}$$

8. A method for detecting an intersection wherein, an intersection between a management object and a protect object is obtained by calculating in each bit a logical product (and) between the "scope[]" obtained in claim 2 or 3 or 4 and the "targets[]" obtained in claim 5.

9. An access control method by using MOI as an access unit comprises; a step for calculating in each bit a logical product (logical and) between a denial of each "targets[]" in an item denial rule obtained in claim 5, which is shown in below equation 2, and the "scope[]" obtained in claim 2 or 3 or 4 or 6,

$$\text{(equation 2)} \qquad \text{denial of "targets[]"=" }\overline{\text{targets[]}}\text{"}$$

and, a step for allowing only an MOI of which scope[i] is 1 based on said calculation.

10. An access control method by using MOI as an access unit comprises; a step for calculating in each bit a logical product (logical and) between each"targets[]" in an item permission rule obtained in claim 5 and the "scope[]" obtained in claim 2 or 3 or 4 or 6, and a step for allowing only an MOI of which scope[i] is 1 based on said calculation.

Fig. 1

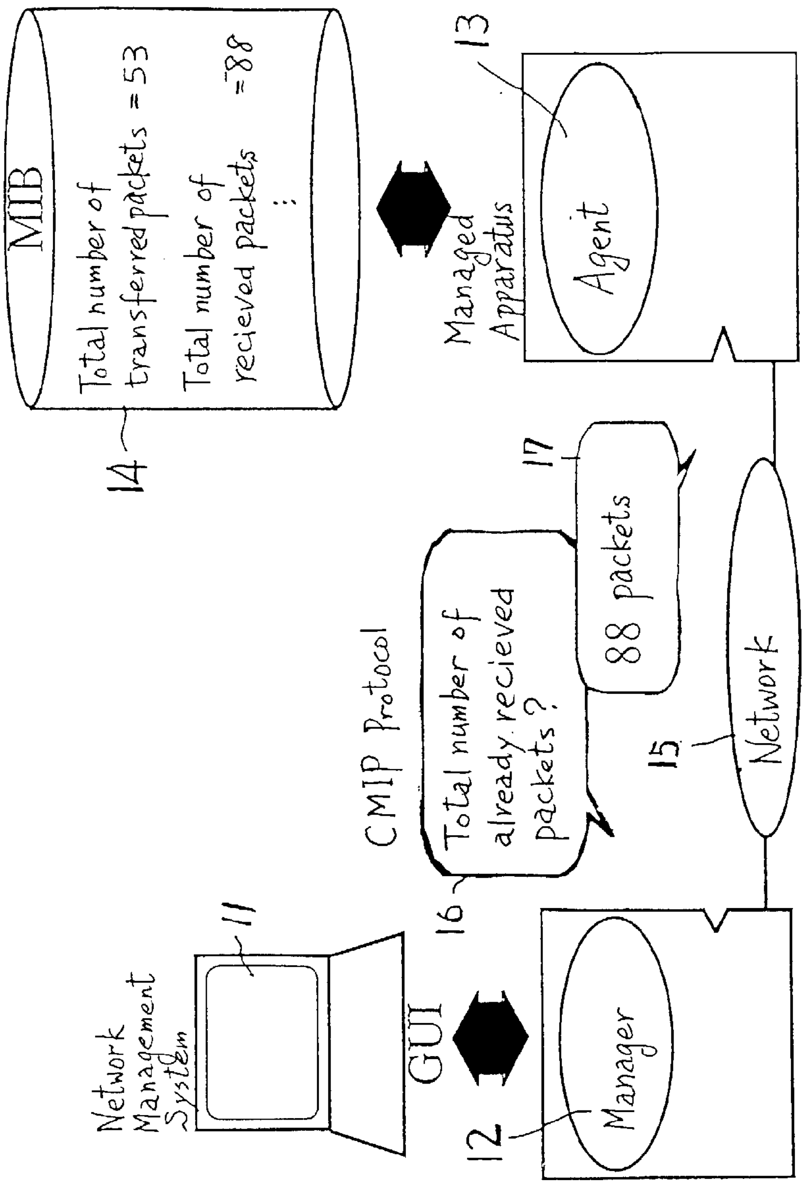

Network Management System
11
GUI
12
Manager
CMIP Protocol
16
Total number of already recieved packets ?
17
88 packets
15
Network
Managed Apparatus
Agent
13
14
MIB
Total number of transferred packets = 53
Total number of recieved packets = 88
⋮

Fig. 2(A)

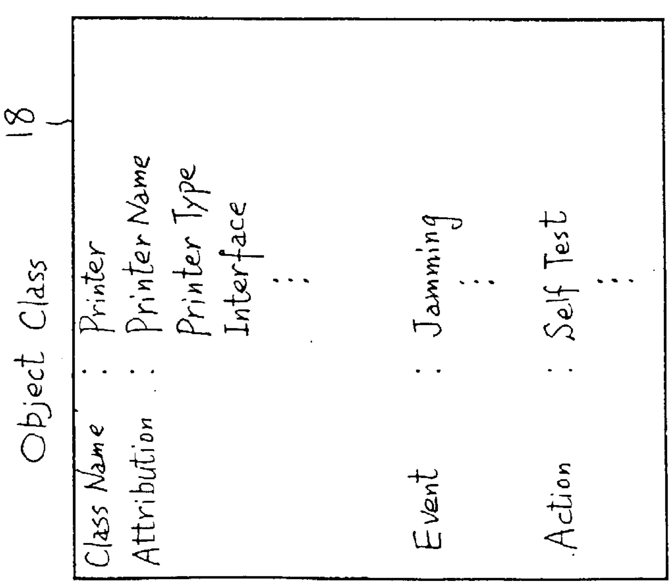

Object Class
18
Class Name : Printer
Attribution : Printer Name
Printer Type
Interface
Event : Jamming
Action : Self Test

Fig. 2(B)

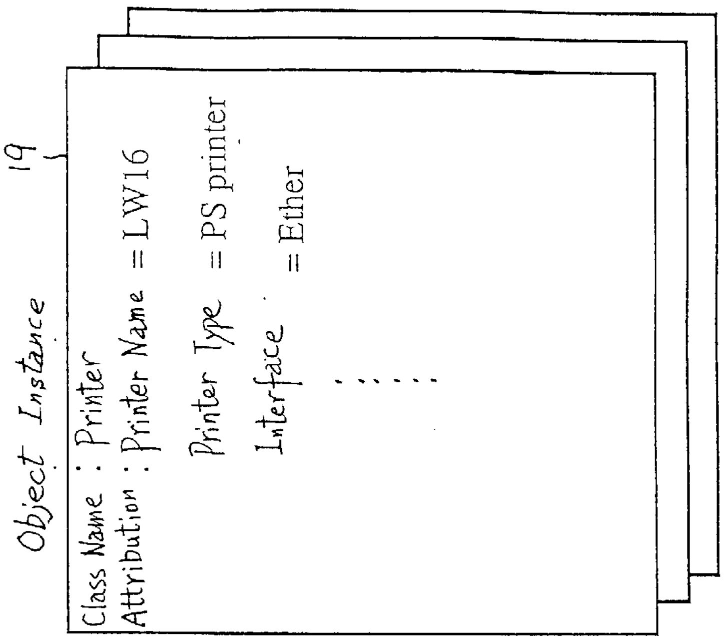

Object Instance
19
Class Name : Printer
Attribution : Printer Name = LW16
Printer Type = PS printer
Interface = Ether

Fig. 3

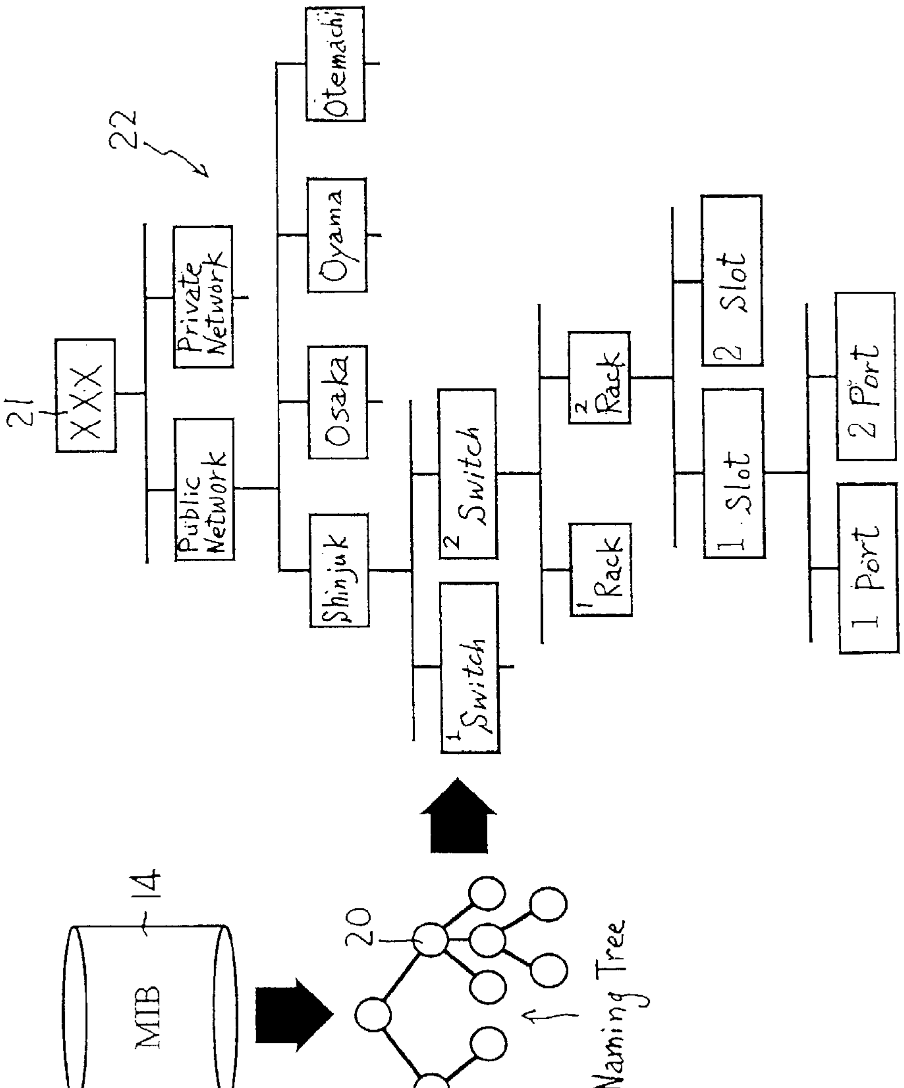
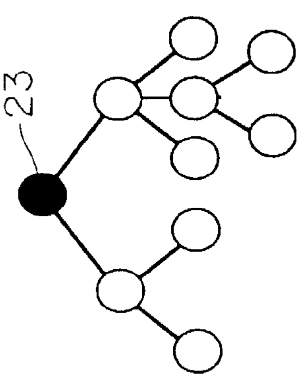
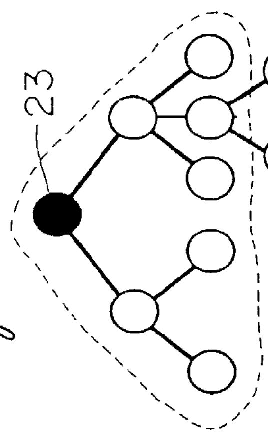

MIB
14
20
Naming Tree
21
XXX
22
Public Network
Private Network
Shinjuk
Osaka
Oyama
Otemachi
1 Switch
2 Switch
1 Rack
2 Rack
1 Slot
2 Slot
1 Port
2 Port

Fig. 4 (A)



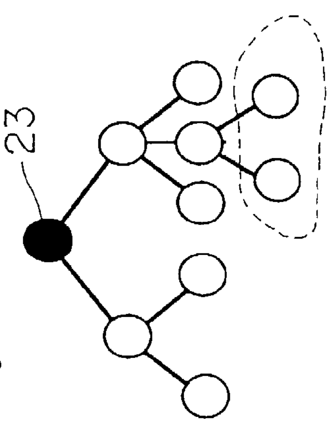
BaseObject

Fig. 4 (B)



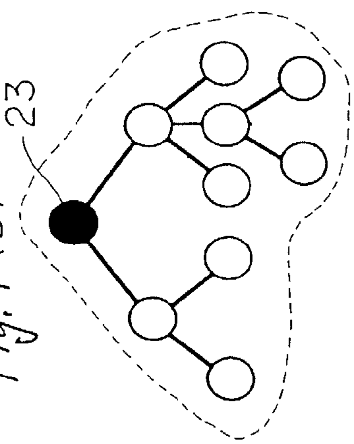
BaseTo2ndLevel

Fig. 4 (C)



3rdLevelOnly

Fig. 4 (D)



WholeSubtree

Fig. 5

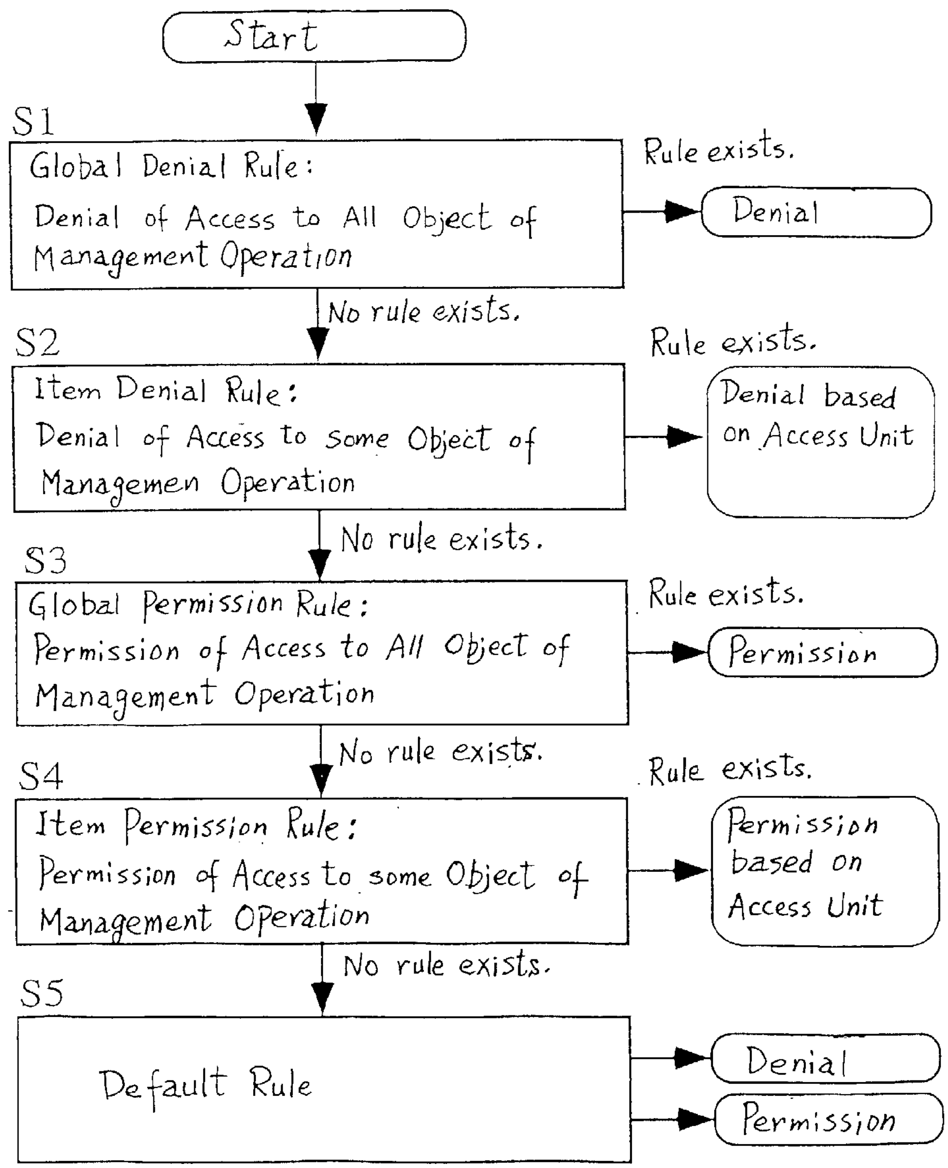

Start
S1
Global Denial Rule:
Denial of Access to All Object of Management Operation
Rule exists.
Denial
No rule exists.
S2
Item Denial Rule:
Denial of Access to some Object of Managemen Operation
Rule exists.
Denial based on Access Unit
No rule exists.
S3
Global Permission Rule:
Permission of Access to All Object of Management Operation
Rule exists.
Permission
No rule exists.
S4
Item Permission Rule:
Permission of Access to some Object of Management Operation
Rule exists.
Permission based on Access Unit
No rule exists.
S5
Default Rule
Denial
Permission

Fig. 6

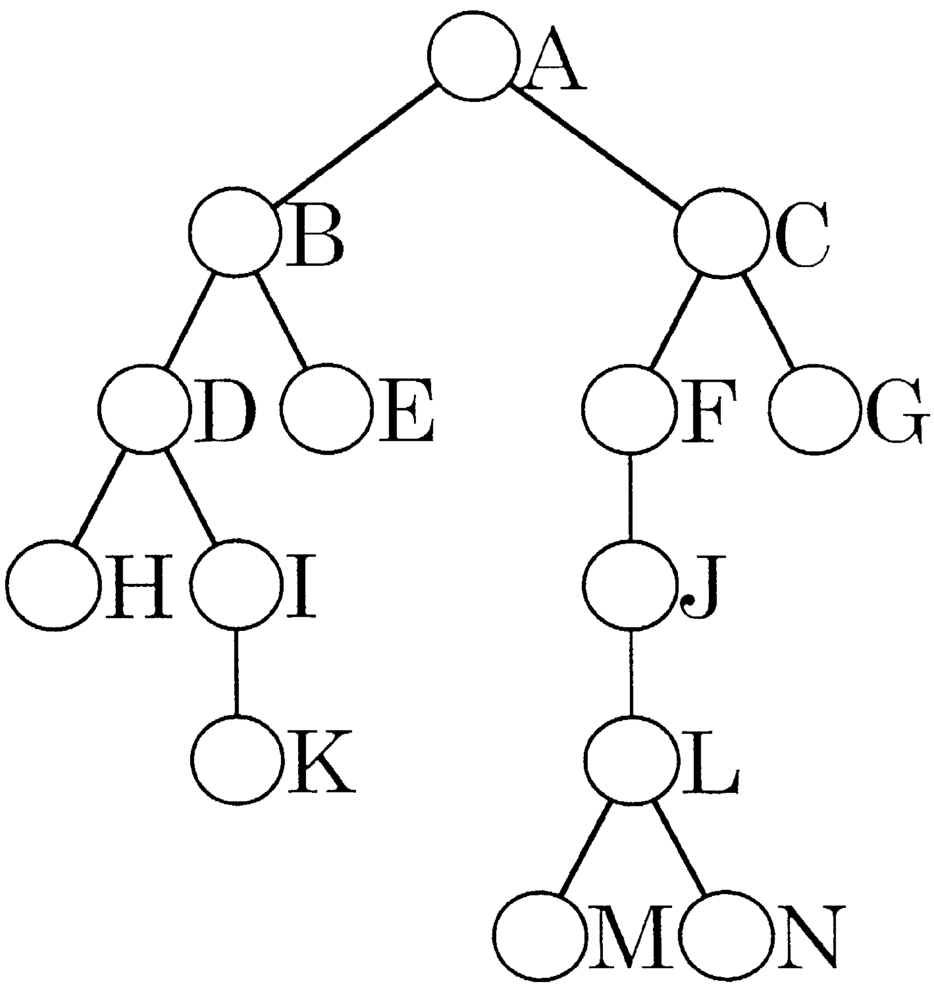


Fig. 7

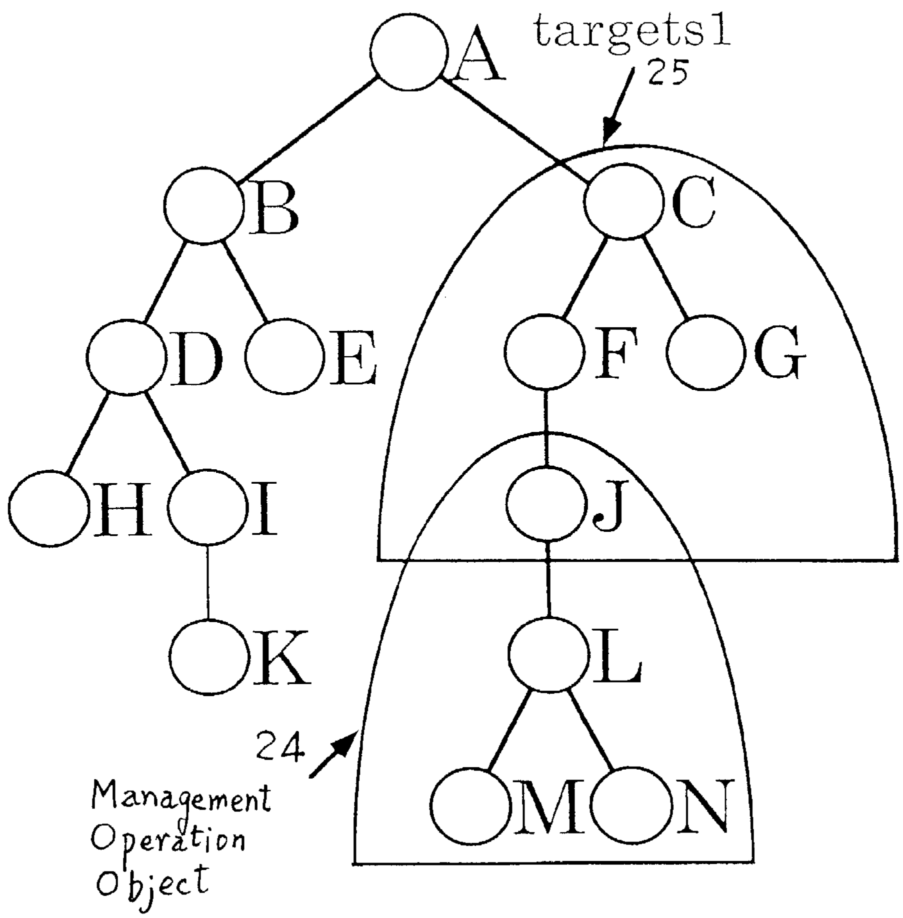

A
targets1
25
B
C
D
E
F
G
H
I
J
K
L
24
Management
Operation
Object
M
N

Fig. 8

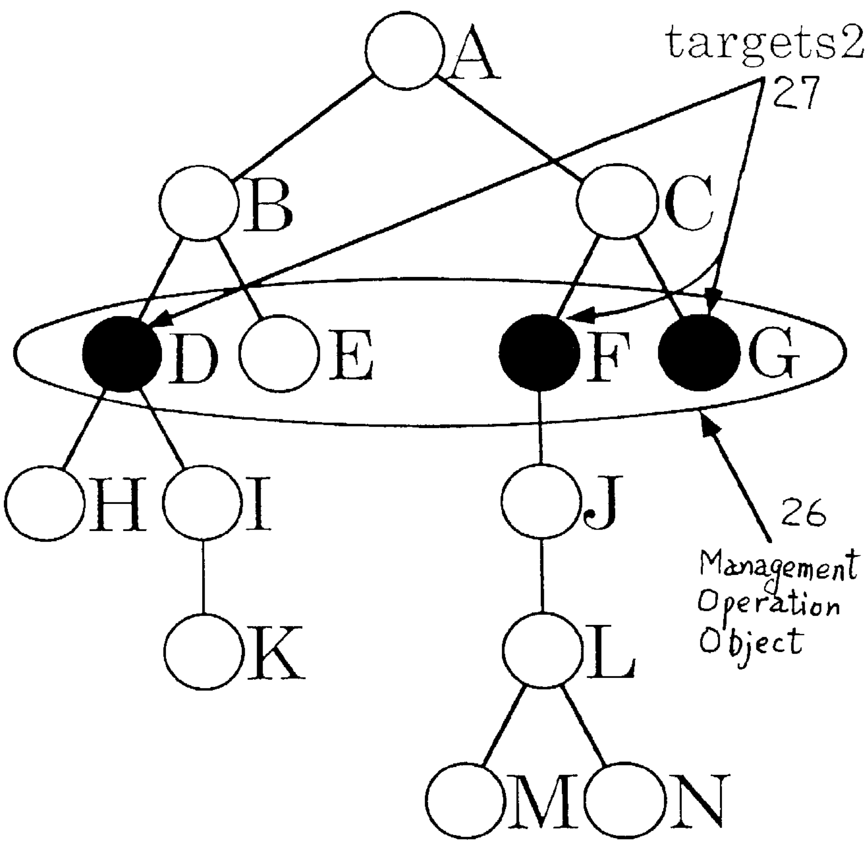

A
targets2
27
B
C
D
E
F
G
H
I
J
K
L
M
N
26
Management
Operation
Object

Fig. 9

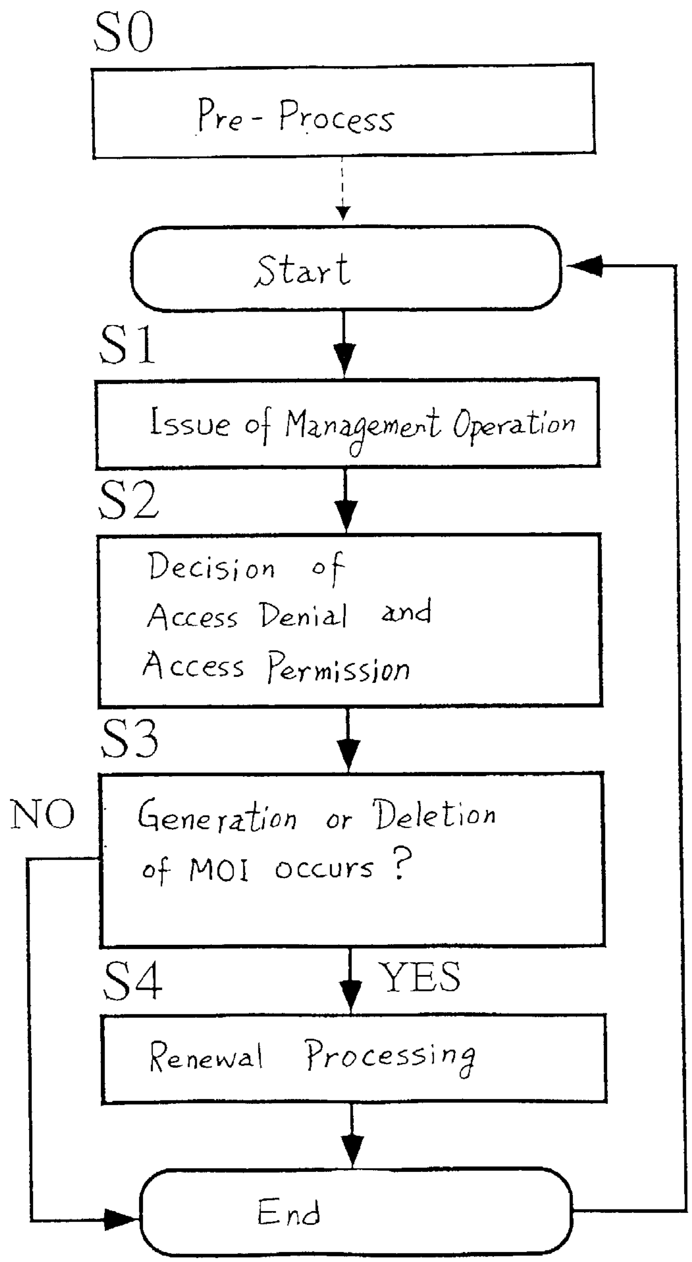

S0
Pre - Process
Start
S1
Issue of Management Operation
S2
Decision of
Access Denial and
Access Permission
S3
NO
Generation or Deletion
of MOI occurs ?
S4
YES
Renewal Processing
End

Fig. 10

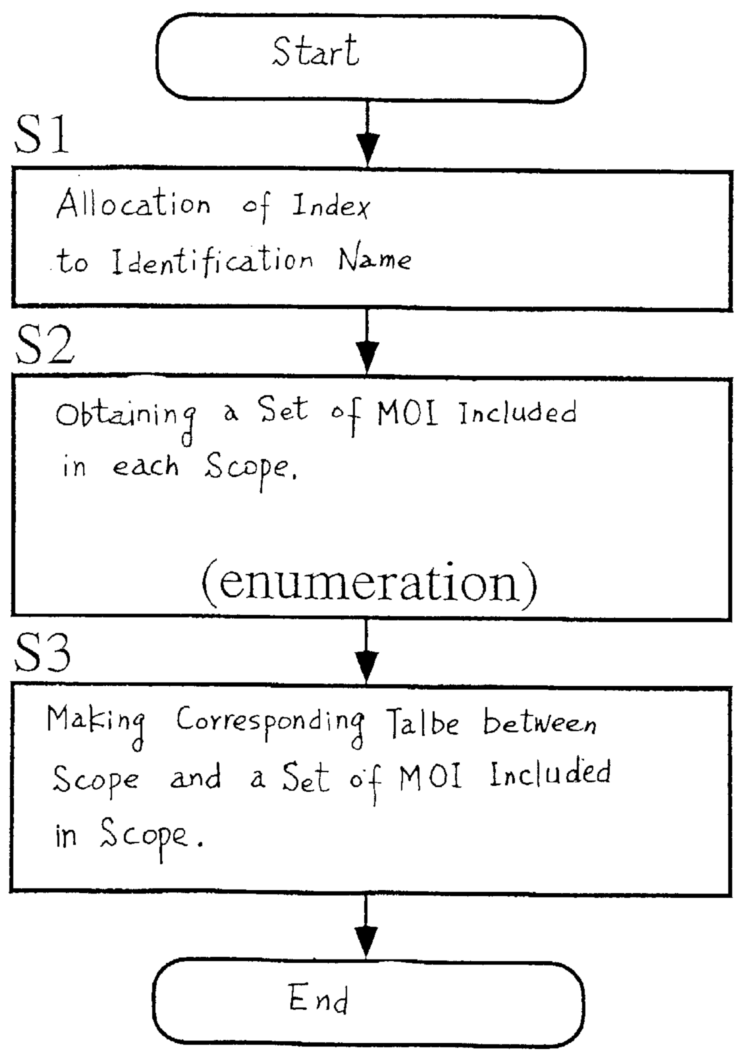

Start
S1
Allocation of Index to Identification Name
S2
Obtaining a Set of MOI Included in each Scope.
(enumeration)
S3
Making Corresponding Talbe between Scope and a Set of MOI Included in Scope.
End

Fig. 11

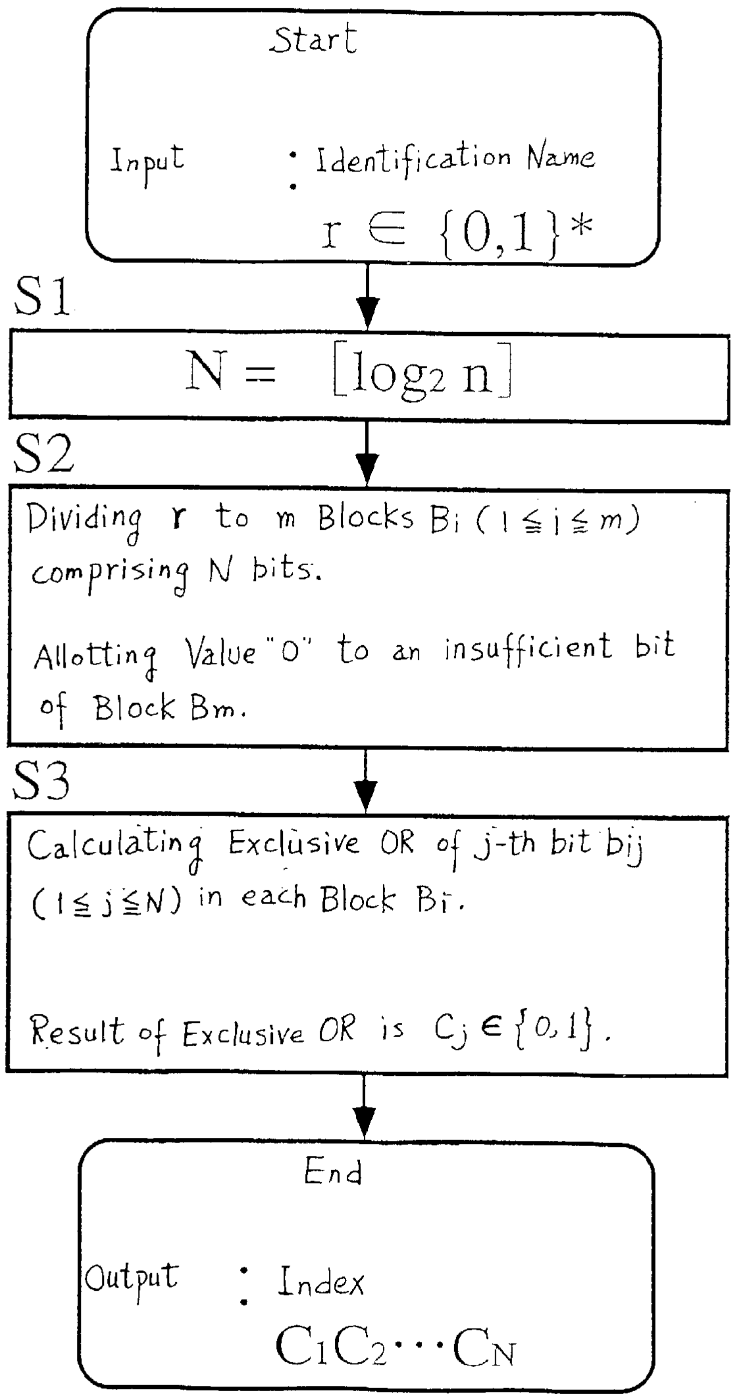

Start
Input : Identification Name
r ∈ {0,1}*
S1
N = [log2 n]
S2
Dividing r to m Blocks Bi (1 ≦ i ≦ m) comprising N bits.
Allotting Value "0" to an insufficient bit of Block Bm.
S3
Calculating Exclusive OR of j-th bit bij (1 ≦ j ≦ N) in each Block Bi.
Result of Exclusive OR is Cj ∈ {0,1}.
End
Output : Index
C1C2···CN

Fig. 12

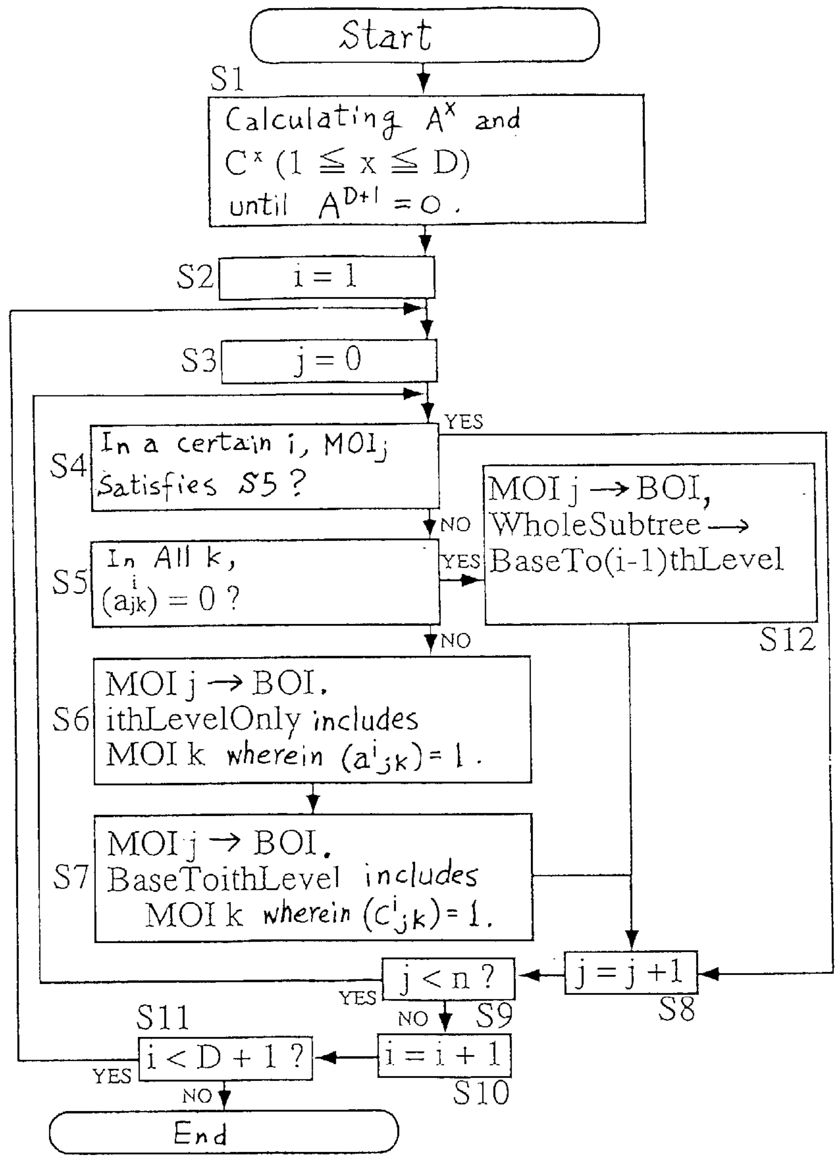

Start
S1
Calculating A^x and C^x (1 ≦ x ≦ D) until A^{D+1} = 0.
S2
i = 1
S3
j = 0
S4
In a certain i, MOI_j Satisfies S5 ?
YES
NO
S5
In All k, (a^i_jk) = 0 ?
YES
NO
MOI j → BOI, WholeSubtree → BaseTo(i-1)thLevel
S12
S6
MOI j → BOI. ithLevelOnly includes MOI k wherein (a^i_jk) = 1.
S7
MOI j → BOI. BaseToithLevel includes MOI k wherein (c^i_jk) = 1.
j = j + 1
S8
j < n ?
YES
NO
S9
i = i + 1
S10
S11
i < D + 1 ?
YES
NO
End

Fig. 13

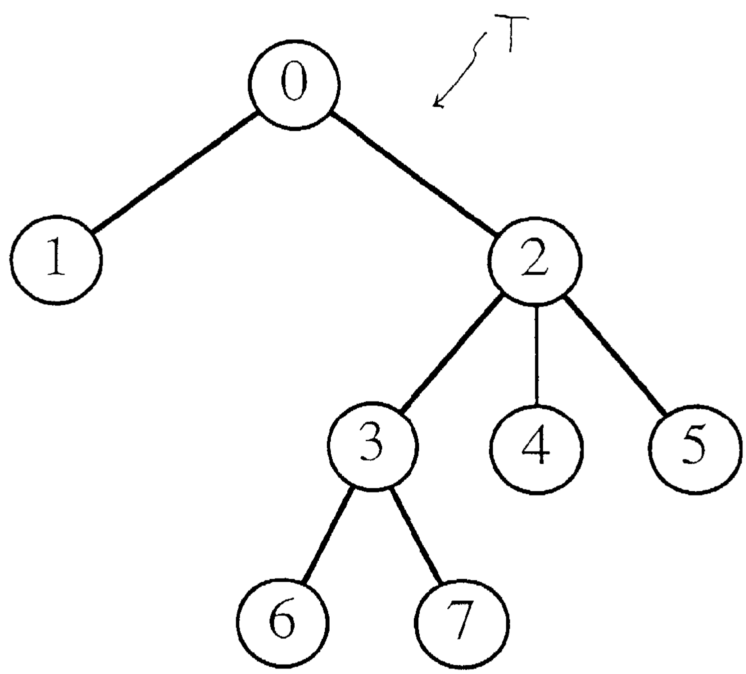


Fig. 14

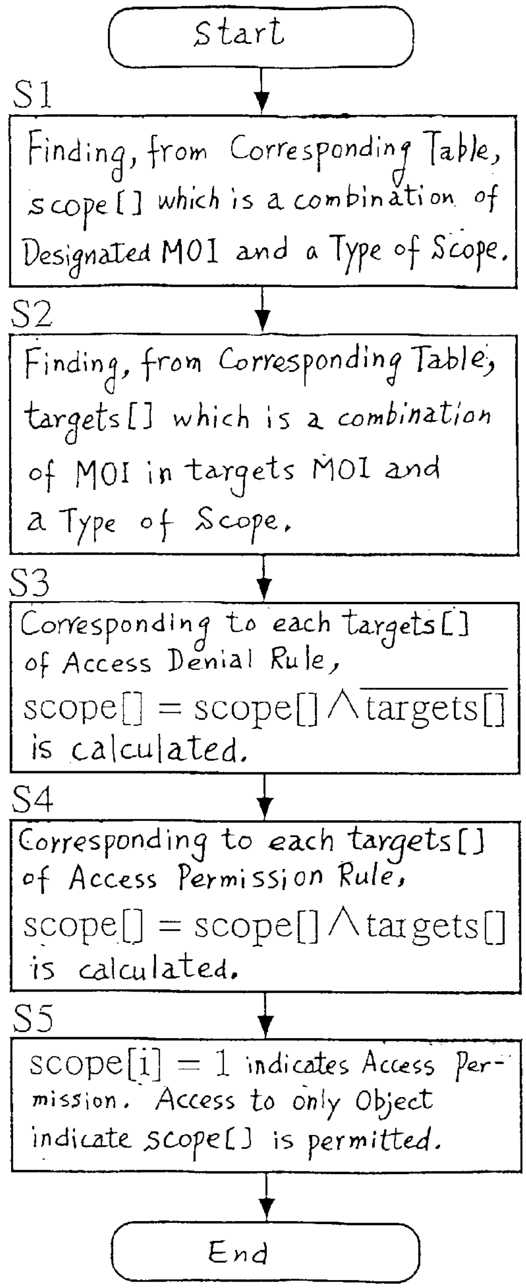

Start
S1
Finding, from Corresponding Table, scope[] which is a combination of Designated MOI and a Type of Scope.
S2
Finding, from Corresponding Table, targets[] which is a combination of MOI in targets MOI and a Type of Scope.
S3
Corresponding to each targets[] of Access Denial Rule, $scope[] = scope[] \wedge \overline{targets[]}$ is calculated.
S4
Corresponding to each targets[] of Access Permission Rule, $scope[] = scope[] \wedge targets[]$ is calculated.
S5
scope[i] = 1 indicates Access Permission. Access to only Object indicate scope[] is permitted.
End

Fig. 15

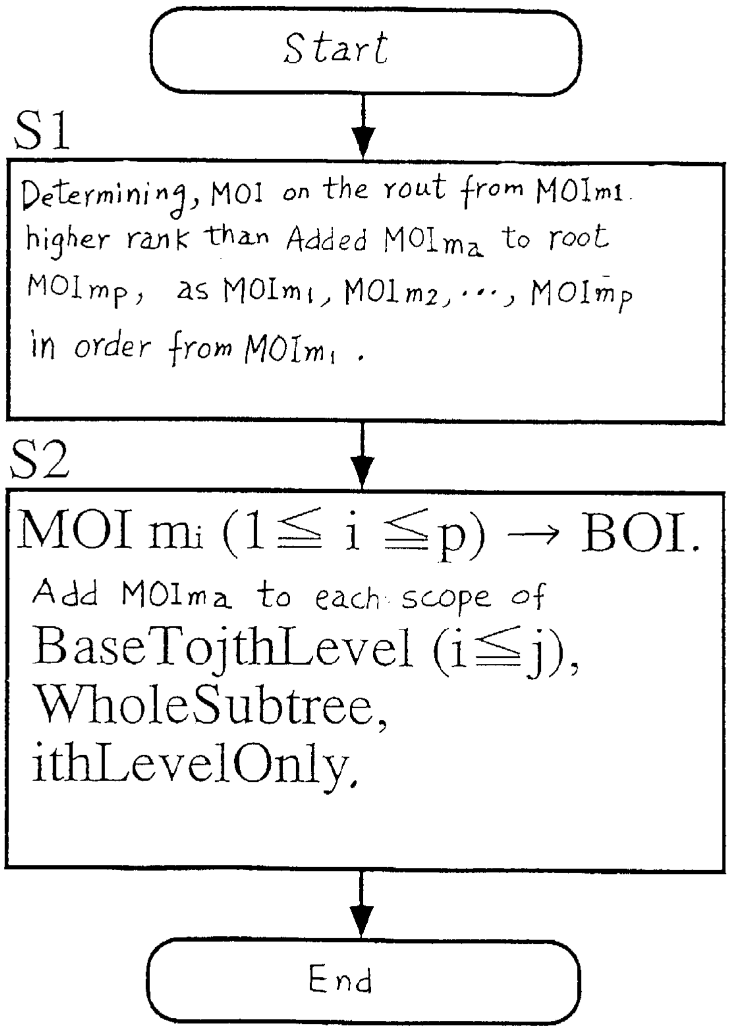

Start
S1
Determining, MOI on the rout from MOIm1 higher rank than Added MOIma to root MOImp, as MOIm1, MOIm2, ···, MOImp in order from MOIm1 .
S2
MOI mi (1≦ i ≦p) → BOI.
Add MOIma to each scope of
BaseTojthLevel (i≦j),
WholeSubtree,
ithLevelOnly.
End

Fig. 16

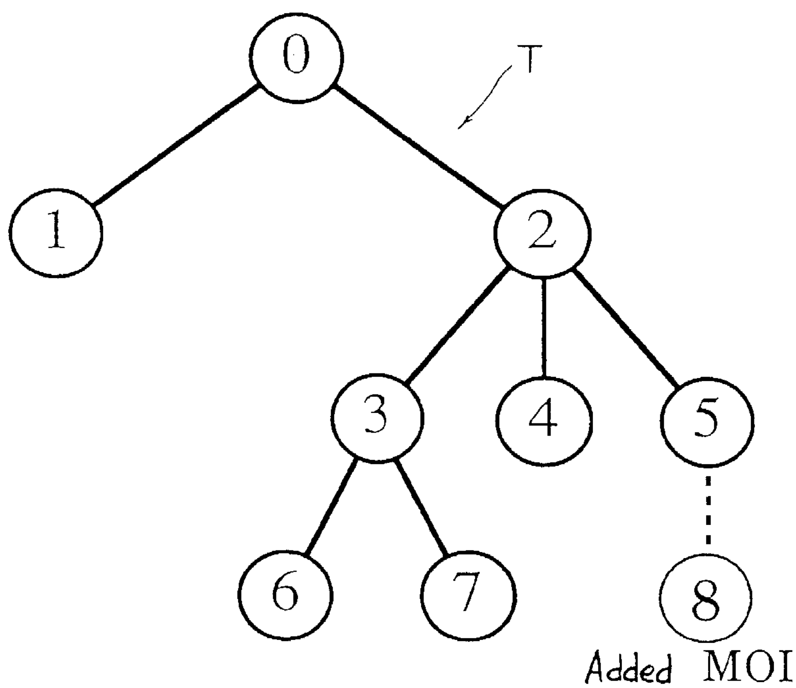


Fig. 17

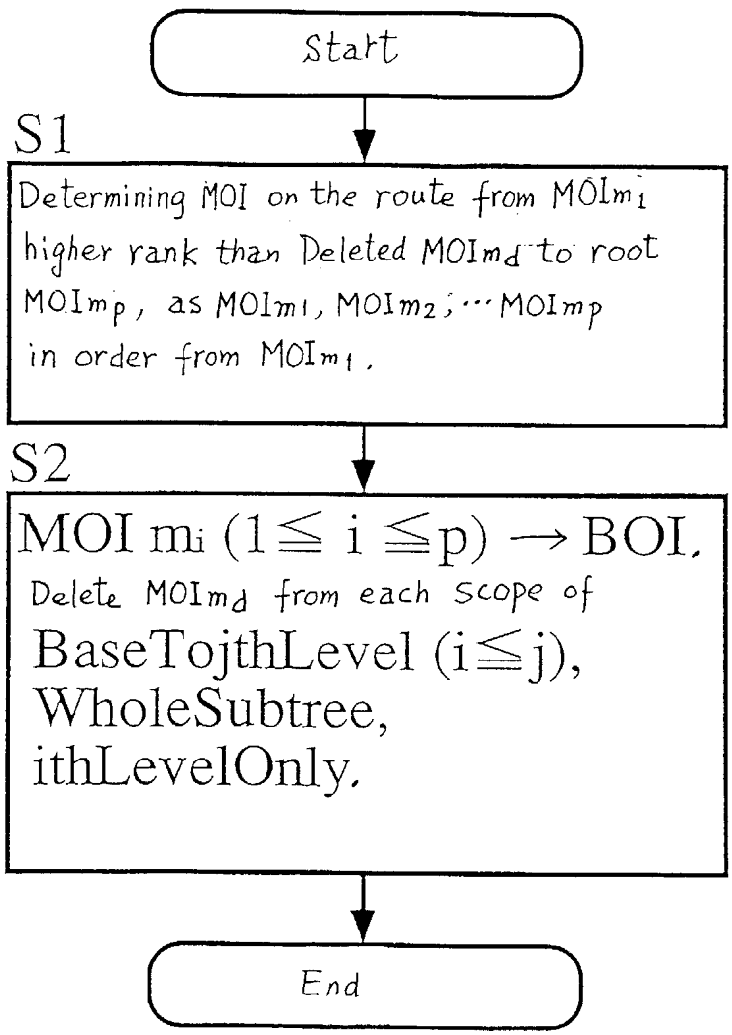

Start
S1
Determining MOI on the route from MOIm1 higher rank than Deleted MOImd to root MOImp, as MOIm1, MOIm2, ··· MOImp in order from MOIm1.
S2
MOI mi (1≦ i ≦p) → BOI.
Delete MOImd from each scope of
BaseTojthLevel (i≦j),
WholeSubtree,
ithLevelOnly.
End

Fig. 18

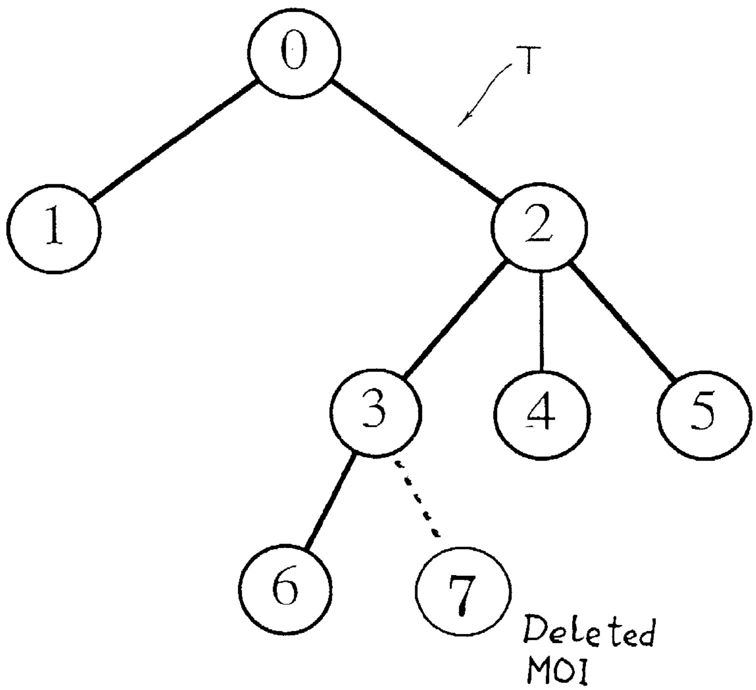


Fig. 19